VIRTUAL BOTTLENECK COMPRESSION WITH FLOW-BASED GENERATIVE MODELS

Elron Bandel
Department of Computer Science
Bar-Ilan University
elronbandel@gmail.com

Lior Frenkel
Department of Electrical Engineering
Bar-Ilan University
liorfrenkel1992@gmail.com

ABSTRACT

Flow-Based models map complex spaces to latent distributions which are easy to sample from. Objects in the complex space can be generated by sampling from this latent distribution followed by a transformation using the inverse of the learned mapping. We propose a method to use such mappings for dimension reduction. We test this method on different data sets and several compression levels, achieving high log likelihood and clear generated images. We compare the results to the Variational Autoencoders (VAE) model [Kingma & Welling (2013)], finding a similar digit outline quality. Hence, this method reduces dimension without harming the invertability and simplicity of the Flow-Based transformation by using a learn-able Virtual Bottleneck. By doing so, this method can preserve the desired invertability of Flow-Based models while simultaneously utilizing the advantages of a low-dimension latent representation.

1 INTRODUCTION

Deep Generative Models like Generative Adversarial Networks (GAN) [Goodfellow et al. (2014)] and Variational Autoencoders (VAE) [Kingma & Welling (2013)] can approximate the sampling process of complex high-dimension distributions. They do so by sampling from easy to sample, low-dimension simple distributions and then transforming those samples to objects of high-dimension distributions. As a side effect of this process, those models learn a meaningful low-dimension representation of the complex space. Flow-Based Deep Generative Models also learn mapping to simpler spaces but with the same number of dimensions as the complex origin. Unfortunately, unlike VAE, Flow-Based models can not use simple dimension reduction, such as matrix multiplication projection [Wold et al. (1987)], These models are restricted to use mapping functions which are invertible. Moreover, in order to make them usable, their Jacobian Determinant should be easy to compute [Dinh et al. (2014)]. Due to these restrictions, many Flow-Based models such as RealNVP [Dinh et al. (2016)] and NICE [Dinh et al. (2014)] presented successful generative models but without the desirable benefits of low dimension latent representation.

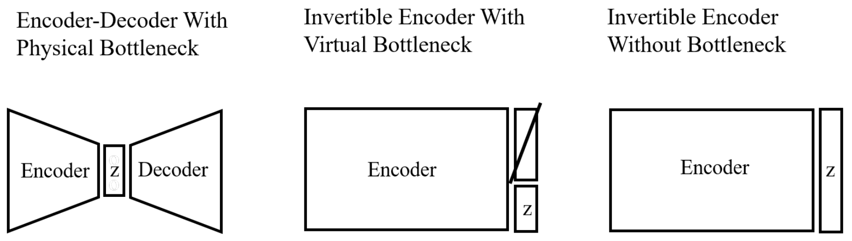

Encoder-Decoder With Physical Bottleneck
Invertible Encoder With Virtual Bottleneck
Invertible Encoder Without Bottleneck
Encoder
z
Decoder
Encoder
z
Encoder
z

Figure 1: Virtual Bottleneck compared to other architectures.

In this work, we present a novel deep learning method of compression by using a learn-able virtual bottleneck. Our proposed bottleneck does not reduce dimensions physically but it reduces the amount of dimensions with new information. We attempt to learn a virtual bottleneck function that produces output with many dimensions that can be thrown away without any loss of information. Plugging our virtual bottleneck at the output layer of any encoder function can reduce the dimension of the output without changing the characteristics of the encoder. We use this virtual bottleneck to learn Flow-Based models that will allow sampling from low dimension distributions for generation of high dimension complex objects.

2 Flow-Based model with Virtual Bottleneck

Flow-Based models estimate a function f that maps complex input space X with probability p_X into a simple probability space Z. If f is invertible, y can be sampled easily from p_Z and can generate an object in the complex space $f^{-1}(z) \in X$. Using the change of the variable formula $p_Z(z) = p_X(f^{-1}(z))|det(J_{f^{-1}}(z)|$, we can calculate the log-likelihood ($\mathcal{L}_{flow}$) of a given dataset $D \subseteq X$ under simple p_Z. Optimizing the parameters of f to a higher likelihood can create a better generative model f^{-1}. NICE is a model that simplifies this process by using Coupling Layers with an easy-to-compute Jacobian determinant. In this work, we use NICE's Additive Coupling Layers with a Jacobian Determinant that equals 1, which makes it simpler and more stable to train.

Virtual Bottleneck Virtual Bottleneck reduces the dimension of the input in the information level, while in practice the output stays with the same dimensions. More formally, $g : \mathbb{R}_d \rightarrow \mathbb{R}_d$ is a Virtual Bottleneck only if $g(x)[1 : t]$ is used. We introduce two such virtual bottlenecks and compare between them in the experiment section.

Null Virtual Bottleneck Null Virtual Bottleneck (NVB) is a virtual bottleneck such that all the entries except one get a value of 0. Thus, if $t-1$ is the upper index of the first slice, we define $g(x)[t :] = 0_{d-t}$. In order to minimize the Mean Square Error (MSE) between $g(x)[t :]$ and 0_{d-t} over the data set, we added it between different slices to our learning objective, in order to allow the NVB model to learn this constraint, $\mathcal{L}_{bottleneck} = \sum_{x \in D} \frac{1}{d-t} \sum_{i \in [0,d]} (g(x)[t :][i] - 0_{d-t}[i])^2$. This architecture is presented in Figure 2.

Redundancy Virtual Bottleneck Redundancy Virtual Bottleneck (RVB) g is a Virtual Bottleneck that produces redundant outputs. More formally, $f(x) = y = Concatenation(y_1, ..., y_n)$ such that $\forall i, j : y_i = y_j$. In order to allow the RVB model to learn the constraint of similarity between all the output chunks, we force our learning objective to minimize $\mathcal{L}_{bottleneck} = \sum_{x \in D} Var(y_1, ...y_n)$. We define the number of copies c as the compression factor of the model. This architecture is presented in Figure 2.

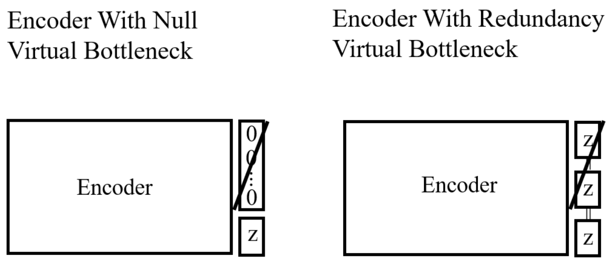

Encoder With Null Virtual Bottleneck
Encoder With Redundancy Virtual Bottleneck
Encoder
0
0
0
z
Encoder
z
z
z

Figure 2: The Architectures of NVB and RVB

In conclusion, our goal is to minimize the virtual bottleneck loss, $\mathcal{L}_{bottleneck}$, and to maximize the log likelihood of the flow model, $\mathcal{L}_{flow}$. Thus, the final objective is to minimize:

$$\mathcal{L} = \mathcal{L}_{bottleneck} - \mathcal{L}_{flow}.$$

Probabilistic interpretation of our objective Another way to approach the combination of the Flow-Based Model with Virtual Bottleneck is by noticing that the only difference that sets it apart from the regular flow models is the target probability space. In the NVB model, we learn a mapping to a simple distribution of concatenated zeros with high probability in the output vector, while in the RVB model we learn a mapping to another simple distribution that produces copies of the same object that was drawn. After sampling from the low-dimension distribution, we generate copies of the sampled vector (RVB) or add zeros (NVB) to fit the input dimension of the decoder. Through this process we can generate a high-dimensional object from a sampled object with a low-dimension distribution by using a small amount of additional information. This indicates that the actual information that is given to the decoder model f is of a lower dimension than its actual input dimension. With this perspective, the simple distribution can be presented as an information-level compression of the complex distribution while having the same actual dimensions.

3 Experiments

In all five experiments, we use the NICE model with one of the two Virtual Bottlenecks described above. The model contains four Additive Coupling Layers and ten MLP layers with Leaky ReLU for each coupling layer. For ease of sampling, we use the log-normal distribution as our target distribution. We refer to this NICE model with NVB or RVB and a compression factor c as NICE-XcNVB or NICE-XcRVB, respectively.

3.1 Comparison of NICE with NVB vs. RVB

We trained a NICE Model consisting of four additive coupling layers with hidden space with a size of 2000 on the MNIST data set for 100 epochs. We trained it once with NVB and once with RVB, both with a compression factor X8.

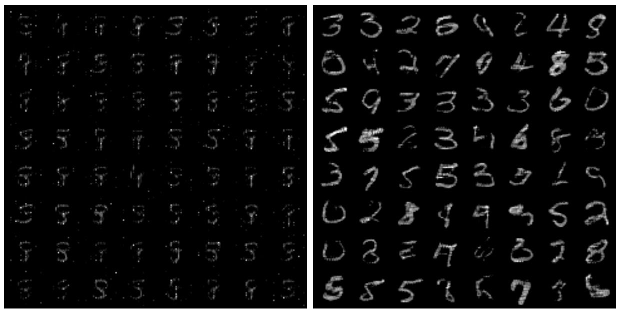

(a) NICE-X8NVB (b) NICE-X8RVB

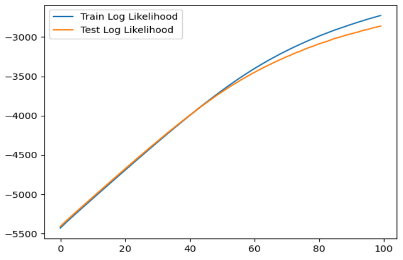
Figure 3: MNIST samples from NICE with X8 Compression RVB/NVB Bottleneck.

Using the best hyper-parameters we found, the images obtained from the RVB model were more clear (Figure 3) and with a higher log likelihood than the images obtained from the RVB model, so we chose to implement the rest of the experiments using RVB and not NVB.

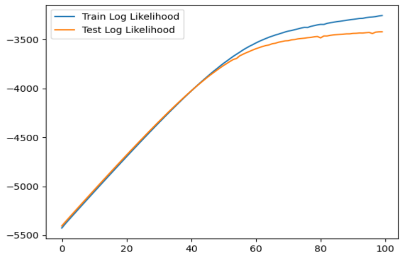
3.2 Comparison of NICE-RVB on different data sets

After choosing to use the RVB model, we also trained it on the fashion-MNIST data set with a compression factor of X8. For both data sets, the model achieved high log likelihood and performed equally on test data sets which were unseen during training (Figure 4).

The images generated by the model trained on the fashion-MNIST data set are not as bold as the images generated from the model trained on the MNIST data set, but they still relatively preserve the characteristics of images in the original data set (Figure 5).

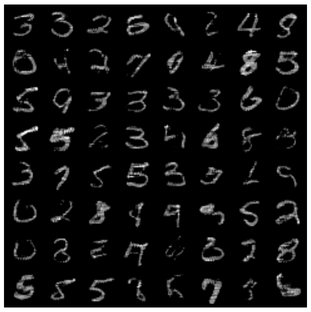


(a) Log likelihood MNIST

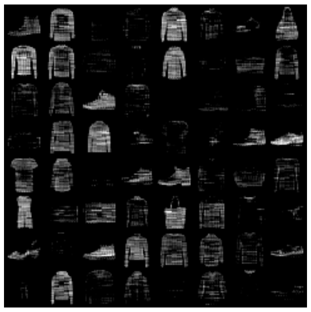


(b) Log likelihood fashion-MNIST

Figure 4: Log likelihood of train and test for NICE-X8RVB trained on (a) MNIST and (b) fashion-MNIST.



(a) MNIST

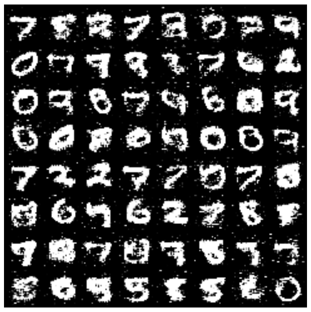


(b) fashion-MNIST

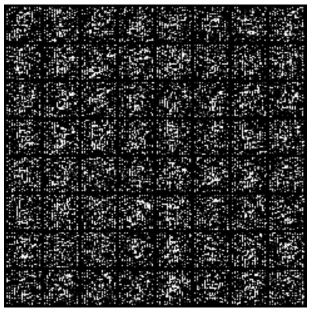
Figure 5: Samples from NICE-X8RVB trained on (a) MNIST and (b) fashion-MNIST.

3.3 Comparison of NICE-RVB with and without the Variance Minimization Objective

We trained our NICE-RVB model with a compression factor of X8 on MNIST for 20 epochs. The model was trained once with Variance Minimization in the loss function and once without it.



(a) With



(b) Without

Figure 6: MNIST samples from NICE-RVB (a) with X8 Compression Bottleneck (b) without Variance Minimization Objective.

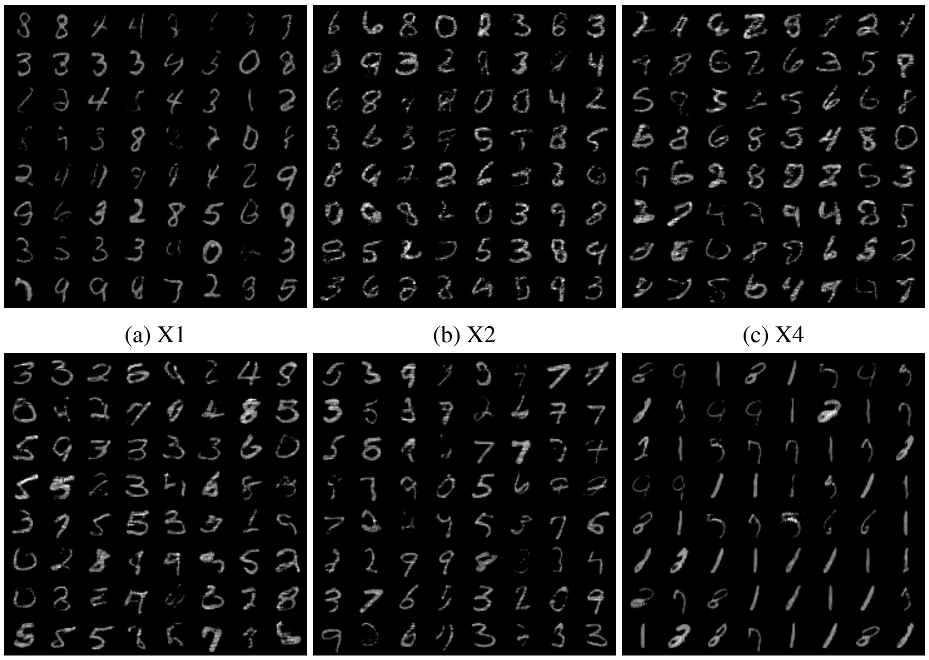
The results show that the Variance Minimization is a necessary component of the Redundancy Bottleneck (Figure 6). Our conclusion is that the similarity between the different copies obtained with the RVB model is essential for creating a sampling process that is equivalent to the training process.

Test Results			
Compression	**Bottleneck # Dim**	**Log likelihood**	**Variance**
X1	784	-5368	0
X2	392	-3004	499
X4	196	-2921	473
X8	98	-2861	391
X16	49	-2822	294
X392	2	-2564	24.33

Table 1: A quantitative comparison between different levels of compression.

3.4 Comparison of NICE-RVB with different compression factors

In the previous experiments, we used a compression factor of X8, i.e., the number of copies obtained from the RVB model was eight. We wanted to compare the results with other levels of compression, so we tested the model with compression factors of X2, X4, X16 and X392. In addition, we also tested the case of no compression (X1) with the standard NICE model.



(a) X1 (b) X2 (c) X4

(d) X8 (e) X16 (f) X392

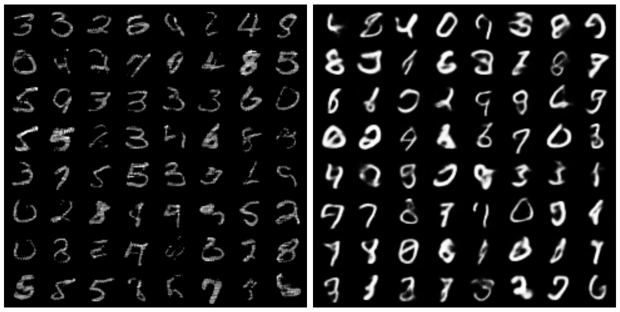
Figure 7: Samples from NICE-RVB trained on MNIST with different compression factors.

Our experiment shows that higher compression factors achieve higher log likelihood (Table 1). At first this might seem unusual, but considering the small dimension it might be easier to achieve high log likelihood on low dimension distributions. We also show that higher compression factors achieve lower variance (Table 1). This is also unexpected because for high compression factors we need to minimize the variance between more copies. We believe this happens because in low dimensions the model needs more similar copies in order to achieve high log likelihood.

The generated visual results seemed to have a similar quality to one another, except for the highest compression factor we tested (X392), which generated images from the same class repeatedly (Figure 7). Besides for the highest compression factor, our method seemed to compress the images without a loss of image quality.

3.5 Qualitative comparison of NICE-RVB with VAE

We trained VAE with a compression factor of X8 (a latent dimension with a size of 98) and compared the generated images with images generated from our NICE-X8RVB model.



(a) NICE-X8RVB (b) X8VAE

Figure 8: MNIST samples from (a) X8NICE-RVB and X8VAE (b) trained on MNIST.

The images generated by the VAE seemed to be bolder, but both models seem to have a similar quality in terms of digit outline (Figure 8). As a compression model, we believe our model is more desirable because it is based on an invertible compression function. Thus, decoding our compressed representation has a deterministic result, while the VAE decodes its compressed representations stochastically.

4 Conclusions

Virtual Bottleneck is an effective method for creating a generative model with a meaningful low-dimension intermediate latent representation. Our method manages to compress images to low dimensions. Unlike stochastic compression methods such as VAE, our deterministic compression method can reliably compress an image to a lower dimension and recover its origin without any loss of information. As a generative model, NICE with a Redundancy Virtual Bottleneck seemed to generate images as effectively as the original NICE model while sampling from a lower dimension latent distribution.

References

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.